**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

#10 6-4-03

**CERTIFICATE OF MAILING**

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Commissioner of Patents and Trademarks, Mail Stop Appeal Brief-Patents PO Box 1450, Alexandria VA 22313-1450 on **May 27, 2003**.

Michael J. Buchenhorner
Name of Person Mailing Papers

Michael J. Buchenhorner
Signature



| | | |
|---|---|---|
| In re Application of | : | Thomas PUZAK |
| Application Number | : | 09/458,883 |
| Attorney Docket Number | : | YOR999-589 |
| Group Art Unit | : | 2183 |
| Filed | : | 12/10/1999 |
| Examiner | : | Wood, William H. |

For: Prefetching Using Future Branch Path Information Derived from Branch Prediction

Commissioner for Patents
Mail Stop Appeal Brief - Patents
PO Box 1450
Alexandria VA 22313-1450

## APPEAL BRIEF

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal filed on March 24, 2003.

## REAL PARTY IN INTEREST

International Business Machines Corporation is the assignee in the above-identified patent application.

## RELATED APPEALS AND INTERFERENCES

The Appellant, the Appellant's legal representative, and the Assignee are not aware of any other appeals or interferences which will directly affect, be directly affected by, or have a bearing on the Board's decision in this Appeal.

## STATUS OF CLAIMS

Claims 1, 3-12, and 14-22 (the claims at issue) are pending in the above-identified patent application. The claims at issue were finally rejected in an Office Action dated December 17, 2002. The final rejection of the claims at issue is hereby appealed.

The claims at issue stand rejected under 35 U.S.C. § 103(a) as being unpatentable over United States Patent No. 5,742,804 issued to Yeh et al (hereafter "Yeh").

## STATUS OF AMENDMENTS

The above-identified patent application was filed on December 10, 1999. A first Office Action was issued on June 25, 2002, rejecting Claims 1-22. On September 25, 2002 an Amendment was filed in response to the first Office Action, wherein claims at issue were amended and claims 2 and 13 were cancelled. A Final Office Action was issued on December 17, 2002 rejecting all pending claims. On March 24, 2003 a Notice of Appeal to the Board of Appeals was filed. Applicant is concurrently herewith transmitting an amendment of claims 21 and 22 changing them to apparatus limitations and their dependency to claim 12 (the structural counterpart to claim 1) to overcome the objection to those claims.

## SUMMARY OF INVENTION

The invention claimed relates to a method for executing an instruction stream comprising the steps of: deriving first path data from a compiler by analyzing control flow information during compilation (page 7, line 20, et seq.), wherein the first path data represents a first path from the prefetch (touch) instruction to an instruction that uses information prefetched by the prefetch instruction (page 7, lines 23-24); obtaining a branch history (mask) defining a path from information generated by branches encountered prior to a subsequent encounter of the prefetch instruction; generating second path data (page 7, line 25, et seq.), wherein the second path data represents a predicted second path of execution; determining whether the first path is consistent with the predicted second path (page 7, lines 27-28); and prefetching instructions and data when the first path is consistent with the predicted second path (page 7, lines 28-31).

## ISSUE

Whether the Examiner erred in rejecting the claims at issue under U.S.C. § 103(a) as being unpatentable over Yeh.

## GROUPING OF CLAIMS

For purposes of this Appeal only, the claims at issue stand or fall together.

## ARGUMENTS

**THE EXAMINER ERRED IN REJECTING THE CLAIMS AT ISSUE UNDER 35 U.S.C. § 103(a) AS BEING UNPATENTABLE OVER THE YEH PATENT.**

The Examiner has erred as a matter of law by failing to establish a prima facie case of obviousness. To reject claims in an application under section 103, an examiner must show an unrebutted prima facie case of obviousness. See In re Deuel, 51 F.3d 1552, 1557, 34 USPQ2d 1210, 1214 (Fed. Cir. 1995). In the absence of a proper *prima facie* case of obviousness, an applicant who complies with the other statutory requirements is entitled to a patent. See In re Oetiker, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992). On appeal an appellant can overcome a rejection by showing insufficient evidence of prima facie obviousness. See id. In this case, the Examiner reached the conclusion of obviousness without substantial evidence supporting the conclusion of obviousness and using incorrect legal standards. The Yeh patent cited in support of the obviousness rejection lacks one of the claimed elements by the Examiner's own admission and also lacks at least one other element that the Examiner erroneously read on the Yeh patent.

THE "OBTAINING A PATH HISTORY" ELEMENT.

Claim 1 requires *inter alia* a step of obtaining a branch history defining a path from information generated by branches encountered prior to a subsequent encounter of the prefetch instruction. It is important to note here that according to the claimed invention the branch history is obtained prior to encountering a prefetch or a branch instruction. The Examiner contends that this element is disclosed at column 6, lines 28-37 of Yeh. That section reads:

"The way that this cancellation policy is implemented in one embodiment of the present invention is by first predicting (using dynamic and static branch predictors) an execution path of the programmed sequence of instructions, and then comparing the execution path with the trace vector for the branch predict instructions. In the event that the execution path and the trace vector do not match, the prefetched request of the associated branch predict instruction is canceled. This avoids bottlenecking branch predict instructions, which can lead to an excessive number of prefetch requests."

Examination of the part of Yeh quoted above reveals that the claim term "branch history" is not found in the passage. In support of the above reading of the claim language the Examiner states: "specifically mentioned is dynamic branch prediction which indicates

branch history of branches long before the prefetch is currently encountered." The entire Yeh patent does not elaborate on specifically what are dynamic branch predictors. Moreover, the Examiner did not construe the claim term "branch history." Construction of claim language is required to reach a conclusion of obviousness. In construing this term, the entire claim language must be considered. The claim elaborates on the meaning of "branch history" by specifying that it defines a path from information generated by the branches encountered *prior* to an encounter with the prefetch instruction. The specification of the application at issue fully supports the branch history. See e.g., Figures 7 and 8 and related discussion. Applicant contends that reading the term "branch history" on the "branch predictor" disclosed in Yeh (at Col. 6, line 30) was error because such a reading is inconsistent with the claim language and the specification.

The Examiner has not shown how the branch predictor of Yeh can correspond to the claimed "branch history." In point of fact it cannot because the path prediction performed in Yeh is done after the prefetch instruction is encountered. The instruction in Yeh that apparently corresponds to a prefetch instruction is at block 10 of Figure 1 of Yeh. See col. 4, lines 49-50.

Because the Yeh patent does not provide any detail on the branch predictor, one must presume that the patent refers to a known branch predictor. There is absolutely no evidence of any branch history being used in Yeh as is claimed by applicants. Moreover, the obviousness rejection provides no secondary reference that teaches a branch history. Therefore, there is no evidence that teaches, motivates or suggests the element of obtaining a branch history. In fact the language of the cited passage in Yeh contradicts the Examiner's rationalization of the reading of the claim. Yeh suggests a comparison of the execution path and trace vector, Col. 6, lines 30-34, and further says: "In the event that the execution path and the trace vector do not match, the prefetched request of the associated branch predict instruction is canceled underlining added]." The use of the word "prefetched" in the past tense means that the instruction was prefetched before the comparison was made. Further evidence of the timing of Yeh is found in the discussion of canceling prefetched instruction if the trace vector does not match the execution path. See col. 6, lines 34-35. Therefore, even if the comparison of the execution patch with the trace vector were to be interpreted to

correspond to obtaining a branch history, comparison of Yeh occurs after the instruction is prefetched. The step of obtaining a branch history by its own terms occurs prior to encountering the prefetch instruction. Therefore, Yeh actually teaches away from the claimed invention.

Further evidence that in Yeh the prefetching occurs before the comparison of the prefetch instruction with the trace vector is found by considering Figure 1 of Yeh. Figure 1 shows the program tree used by Yeh in describing his prefetch mechanism. As shown in the figure and discussed in the related text, Yeh first encounters the prefetch instruction at block 10 and then predicts the execution path to the target instruction (the one to be prefetched). See e.g. offset 21. By contrast, in the claimed method, the branch history is obtained before comparing the predicted patch with the first path (from the prefetch instruction to the instruction to be prefetched). Therefore, applicant's invention would occur before block 10 of Figure 1 of Yeh. Yeh denotes that part as "program flow" but does not discuss what occurs before block 10. Therefore, there is no evidence that supports the Examiner's conclusions of obviousness.

Yet further evidence that in Yeh prefetching occurs before comparing paths is found in the claims of Yeh. Thus, in claim 1 step (b) requires prefetching a block of instructions. It is not until claim 7 that a comparison of paths is made. Claim 7 depends on claim 6 which in turn depends on claim 1. The logical order of the steps compels a conclusion that Yeh must prefetch before comparing paths.

Moreover, the claimed determination of whether the first path and the second path are consistent does is neither taught nor suggested by Yeh. The only path comparison performed in Yeh compares the execution path with the trace vector. The Examiner alleges that the trace vector corresponds to claimed first path. Assuming that this is correct, the comparison is different from the "determining" step because the "predicted second path" of claim 1 is different from the predicted path of Yeh. The term "second path" as used in the application at issue is determined branch history which is determined at a point prior to the encounter with the prefetch instruction. As discussed above, the prediction made in Yeh occurs after encountering the prefetch or predict instruction.

PREFETCHING INSTRUCTIONS AND DATA

The Examiner further erred in concluding that it would have been obvious to modify Yeh such that it prefetches both instructions and data as required by the last step in claim 1 of the patent application at issue in this appeal. The Examiner concedes that "Yeh did not explicitly state prefetching data." However, the examiner argued that Yeh alluded to the concept of prefetching data as well as instructions in col. 1, lines 22-25. The cited portion is the beginning of the "Background of the Invention." It states: "As the operating frequencies of microprocessors continues to rise, performance often depends upon providing a continual stream of instructions and data in accordance with the computer program that is running." This is a very general statement that is made in isolation from the discussion of the invention of Yeh. That statement makes no suggestion of prefetching both instructions and data and in fact Yeh contains absolutely no discussion or disclosure of any mechanism capable of prefetching data. Again, we have nothing except the Examiner's unsupported conclusion that prefetching data is a logical consequence of prefetching instructions. However, such conclusory statements unsupported by substantial evidence of the desirability, motivation or teaching of making the modification cannot as a matter of law be the basis for a conclusion of obviousness. In re Zurko, 258 F.3d 1379 (Fed. Cir. 2001). Deficiencies of cited references cannot be remedied by the Board's [or Examiner's] general conclusions about what is "basic knowledge" or "common sense" to one of ordinary skill in the art. Id.

The Examiner's rationalization of the rejection is equally suspect as the conclusion. Thus, the Examiner states: "Clearly, if the instructions are being prefetched, the data for those instructions are needed to keep up with the continual stream of operation." The Examiner is wrong. There are many computer architectures that prefetch instructions but do not prefetch data. Yeh is one of them. There exist logical reasons not to prefetch data as well as instructions. For one, if the prefetch instruction misses, there is a penalty for prefetching an instruction not executed and data not used. When applicant's claim 1 is considered as a whole, as it must be, it is apparent that the claim requires more than just prefetching instructions and data but it requires the entire claimed combination that includes obtaining a branch history. Thus, because the prefetch instruction is not executed until the

claimed determination of consistency is done, there is no penalty for the miss. Thus, the prefetching of data in the context of the entire claim has been ignored in the rejection.

The Examiner's reasoning is further flawed. Thus, the Examiner says: "It would not make sense to have instructions prepared for execution in a cache and then have those same instructions wait for data." Whether it makes sense or not in the Examiner's opinion, that is precisely what happens in many prior art processor architectures where an instruction is issued for execution and an operand miss occurs and the system waits for retrieval of the correct data. In any case the Examiner's subjective opinion of what makes or does not make sense is not the test for obviousness. The correct test is whether the invention as a whole would have been obvious for one of ordinary skill in the art. If the alternatives to the claimed invention did not "make sense" at the time of the invention, the Examiner bears the burden of proof to establish that fact by substantial evidence. In this case we have only the Examiner's hindsight-driven logic and sense as a substitute for substantial evidence. It is also important to note that when Yeh determines that the trace vector does not match the execution path Yeh cancels the prefetched instruction. If as the Examiner contends Yeh had prefetched data as well there would be some indication of what the processor would do with that prefetched data. The fact that Yeh says nothing about this supposedly prefetched data contradicts the Examiner's conclusion.

The Examiner further states: "If the compiler knows what instructions are going to be executed it *can* also make a guess as to what those instructions will need [emphasis added]." This logic is nothing more than an application of the improper "obvious to try" test of obviousness. However, that is not the standard of 35 U.S.C. §103. In re Geiger, 815 F.2d 868, 2 USPQ2d 1276 (Fed. Cir. 1987). What the processor can or cannot do is not relevant to the obviousness inquiry. That is an open ended inquiry that would be far too open to speculation to serve usefully. Again, the test is what *would* a reasonably skilled artisan have found obvious.

The Examiner attempts to justify the conclusion of obviousness by arguing that Yeh indicates the ability to prefetch instructions and data but instead of providing any teaching or suggestion in Yeh, the Examiner injects his own flawed reasoning that prefetching instructions makes no sense unless the data is also prefetched. As indicated herein, whether

it makes sense or not, the practice of prefetching instructions without prefetching the operand data is common and evidence that this is done in prior art systems is found in Yeh itself. The Examiner does not suggest that the location of the data to be prefetched coincides with the address of the instructions to be prefetched. Moreover, the Examiner has not shown any mechanism whatsoever in Yeh that is capable of prefetching data. The Examiner's logic that prefetching instructions implies prefetching data is simply unsupported by the evidence of record and there cannot be a conclusion of obviousness without some such evidence. Therefore, the Examiner erred in concluding that the claimed invention would have been obvious and the applicants respectfully request reversal of the final rejection.

THE DEPENDENT METHOD CLAIMS AND THE APPARATUS CLAIMS

Claim 12 is an apparatus counterpart of claim 1 and hence is patentable for the reasons discussed above with respect to claim 1. Claims 3-11 depend on claim 1 and hence are patentable for the reasons discussed above.

The Examiner further erred in rejecting claim 4 because the rejection ignores the fact that in the claimed the step of determining whether the first path is consistent with the second path occurs before the prefetching step. As discussed above, in Yeh the instructions are prefetched before the execution path is determined and hence the comparison of the trace vector with the execution path occurs after prefetching.

The Examiner further erred in rejecting claim 6. Although the Examiner conceded that Yeh does not state that the second path data comprises a mask that represents a predicted path of execution that follows branch instructions, the Examiner contends that it would have been obvious for one skilled in the art to perform this step because "it would have made the comparison operation easier." There is absolutely no evidence of this contention. Without such evidence a conclusion of obviousness cannot be sustained. See In re Zurko, supra.

The Examiner further erred in rejecting claim 8. Claim 8 requires that the second path data are based upon accumulation of predictions associated with branch instructions. The Examiner cites Yeh col. 6, lines 13-18. That section discusses the trace vector (which the Examiner contends corresponds to the first path). It does not discuss any path that corresponds to the second (predicted) path of claim 1, let alone as further limited by claim 8.

The cited passage discusses a branch prediction unit of a microprocessor but does not provide any detail on how that is done. Without such evidence the process of determining obviousness *vel non* of an invention becomes a matter of hindsight-guided conjecture and that is not the test for obviousness. Obviousness may not be established using hindsight. See W.L. Gore & Assocs., Inc. v. Garlock, Inc., 721 F.2d 1540, 1551, 220 USPQ 303, 312-13 (Fed. Cir. 1983). The test for obviousness of an invention is meant to be objective. Hence, the test is not a question of whether the invention was or would have been obvious to the inventor or anyone other than the hypothetical of ordinary skill in the art. This is a difficult test to administer because it requires a determination of what those skilled in the art would have considered obvious at the time of the invention. That requires that certain legal standards be followed. In this case, the Examiner has not followed those standards by ignoring the claims as a whole (e.g., the timing of prefetching), the prior art as a whole (Yeh's teachings away from prefetching data), failing to construe the claims consistently with the claim language and the specification (the branch history) and reading conclusions based in hindsight reasoning.

The Examiner further erred in the rejection of claims 11 and 22 by basing the conclusion of obviousness on the supposed "allusion" to the concept of prefetching data and on the Examiner's subjective opinion on that it would not "make sense" to prefetch instructions and then have those instructions wait for the operand data. Yet as discussed above, that is precisely what Yeh does because it teaches no mechanism for prefetching operand data! Teachings away from the invention in the prior art must be considered. A prior patent must be considered in its entirety, i.e., as a whole, including portions that would lead away from the invention in suit. W. L. Gore & Assoc., Inc. v. Garlock, Inc., 721 F.2d 1540, 1550, 220 USPQ 303, 311 (Fed. Cir. 1983), cert. denied 469 U.S. 851 (1984). In this case the Examiner erred in ignoring the teachings in Yeh that lead away from the claimed invention.

## CONCLUSION

In view of the foregoing, it is respectfully submitted that the application and the claims are in condition for allowance. Reversal of the final rejection, and allowance of the claims as amended, are requested.

Respectfully submitted,

Date: May 26, 2003

By: Michael J. Buchenhorner

Michael J. Buchenhorner
Registration No. 33,162
Attorney for Appellant

## APPENDIX

1. (Once amended) In a system including a high speed buffer logically placed between memory and at least one processor unit, a method for executing an instruction stream stored in the memory, wherein the instruction stream comprises a sequence of instructions including at least one prefetch instruction that prefetches information from the memory into the high speed buffer, the method comprising the steps of:

deriving first path data from a compiler by analyzing control flow information during compilation, wherein the first path data represents a first path from the prefetch instruction to an instruction that uses information prefetched by the prefetch instruction;

obtaining a branch history defining a path from information generated by branches encountered prior to a subsequent encounter of the prefetch instruction;

generating second path data, wherein the second path data represents a predicted second path of execution;

determining whether the first path is consistent with the predicted second path; and

prefetching instructions and data when the first path is consistent with the predicted second path.

2. (Cancelled) The method of claim 1, wherein the first path data is derived from a compiler performing static compilation.

3. The method of claim 1, wherein the second path data is derived from information characterizing dynamic execution of the sequence of instructions by the at least one processor unit.

4. The method of claim 1, wherein the prefetch instruction is added to the instruction stream for

execution by the at least one processor unit upon determining that the first path falls within the predicted second path.

5. (Amended) The method of claim 1, further comprising the step of:

upon determining that the first path does not fall within the predicted second path, omitting the prefetch instruction from the instruction stream executed by the at least one processor unit.

6. The method of claim 1, wherein the second path data are associated with one or more branch instructions, and wherein the second path data comprises a mask that represents a predicted path of execution that follows the associated branch instructions.

7. (Amended) The method of claim 6, wherein the first path data comprises a mask that represents a path of execution from the prefetch instruction to the instruction that uses the information prefetched by the prefetch instruction.

8. The method of claim 7, wherein the second path data are based upon accumulation of predictions associated with branch instructions.

9. The method of claim 8 wherein the second path data is derived from predictions based upon previous execution of the instruction stream.

10. The method of claim 1, wherein the prefetch instruction includes a field that identifies an instruction to prefetch from memory into the high speed buffer.

11. The method of claim 1, wherein the prefetch instruction includes a field that identifies data to prefetch from memory into the high speed buffer, wherein the data is operated on by at least one instruction in the instruction stream.

12. (Amended) In a system including a memory storing an instruction stream comprising a sequence of instructions including at least one prefetch instruction, a processor unit for executing the sequence of instructions, and a high speed buffer logically placed between the memory and the at least one processor unit, an apparatus for conditionally executing the prefetch instruction

comprising:

decode logic for deriving first path data from a compiler performing static compilation, wherein the first path data represents a first path from the prefetch instruction to an instruction that uses information prefetched by the prefetch instruction;

logic for obtaining a branch history defining a path from information generated by branches encountered prior to a subsequent encounter of the prefetch instruction;

path prediction logic for generating second path data, wherein the second path data represents a predicted second path of execution;

compare logic for determining whether the first path is consistent with the predicted second path; and

execution logic for conditionally prefetching instructions and data when the first path is consistent with the predicted second path.

13. (Cancelled) The apparatus of claim 12, wherein the first path data is derived from a compiler performing static compilation.

14. The apparatus of claim 12, wherein the second path data is derived from information characterizing dynamic execution of the sequence of instructions by the at least one processor unit.

15. The apparatus of claim 12, wherein the execution logic executes the prefetch instruction
upon determining that the first path falls within the predicted second path.

16. The apparatus of claim 12, wherein the execution logic omits execution of the prefetch instruction upon determining that the first paths does not fall within the predicted second path.

17. The apparatus of claim 12, wherein the second path data are associated with one or more branch instructions, and wherein the second path data comprises a mask that represents a predicted path of execution that follows the associated branch instructions.

18. The apparatus of claim 17, wherein the first path data comprises a mask that represents path of execution from the prefetch instruction to the instruction that uses the information prefetched by the prefetch instruction.

19. The apparatus of claim 18, further comprising branch prediction logic for generating predictions associated with branch instructions, and a branch history queue for accumulating the predictions generated by the branch prediction logic, wherein the second path data generated by the path prediction logic is based upon the predictions accumulated in the branch history queue.

20. The apparatus of claim 19, wherein the predictions are based upon previous execution of the instruction stream.

21. The method of claim 1, wherein the prefetch instruction includes a field that identifies an instruction to prefetch from memory into the high speed buffer.

22. The method of claim 11, wherein the prefetch instruction includes a field that identifies data to prefetch from memory into the high speed buffer, wherein the data is operated on by at least one instruction in the instruction stream.